UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/754,890	01/05/2001	K. Rustan M. Leino	9772-0274-999	4747

7590 05/17/2004

PENNIE & EDMONDS LLP
3300 Hillview Avenue
Palo Alto, CA 94304

EXAMINER	
YIGDALL, MICHAEL J	
ART UNIT	**PAPER NUMBER**
2122	8

DATE MAILED: 05/17/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

7

Office Action Summary	Application No. 09/754,890	Applicant(s) LEINO ET AL.	
	Examiner Michael J. Yigdall	Art Unit 2122	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 19 February 2004.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-47 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-47 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is in reply to Applicant's response and amendment dated February 19, 2004. Claims 1-47 are pending.

Claim Objections

2. Claims 11, 17 and 39 (original) are objected to, with regard to the new listing of claims filed February 19, 2004, because of typographical errors in the recited notations for the flow control label names. Claims 11, 17 and 39 have been treated based on the original listing of claims filed January 5, 2001.

3. Claim 47 (new) is objected to because of possible typographical errors in the recited notations for the flow control label names.

Claim Rejections - 35 USC § 112

4. The rejection of claims 34 and 35 under 35 U.S.C. 112, second paragraph, is withdrawn in view of Applicant's amendments.

Response to Arguments

5. Applicant's arguments have been fully considered but they are not persuasive.

Applicant contends that the combination of Detlefs and Chan does not teach a static checking method that includes flow control labels that identify conditional branch points, and suggests that instead, such a combination would teach a dynamic checker that reports the location of specific error messages at runtime (see page 10 and page 11, top).

However, Detlefs clearly teaches a static checking method, as is shown by the title ("Extended Static Checking"). Detlefs further teaches labels that identify source positions (see page 29, section 6, paragraph 2, lines 7-8). Thus, Detlefs teaches a static checking method that includes labels that identify source positions. Chan teaches markers associated with conditional branches that are used to identify a path through the computer program (see column 1, lines 38-53). The markers taught by Chan thus serve as control flow labels. Therefore, in combination, Detlefs and Chan teach a static checking method that includes control flow labels that identify conditional branch points.

Applicant also contends that no motivation, teaching, or suggestion exists for combining Detlefs and Chan (see page 12). Although the error checking means taught by Chan includes computing a marker at runtime, this marker is "matched with the stored marker to detect any wild branches" (emphasis added; see column 1, lines 41-43). Moreover, the sequence identifier generated along a path at runtime is compared "with the global label stored earlier" (emphasis added; see column 1, lines 54-57). This suggests that at least part of the error checking method taught by Chan involving the markers or flow control labels occurs prior to execution time, as is the case with static checkers.

The Detlefs and Chan references are both directed to error checking. Detlefs teaches a static checking method, as described above. Chan teaches a method having flow control labels, also as described above, for verifying the paths and conditional branches in a program (see column 1, lines 26-35). Therefore, one of ordinary skill in the art would be motivated to combine the static checking method taught by Detlefs with the flow control labels taught by Chan in order to verify the paths and conditional branches in a program.

6. In response to Applicant's arguments against the references individually (see page 11), one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

Claim Rejections - 35 USC § 103

7. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.

8. Claims 1-5, 8, 11, 14, 17, 20, 22-27, 30, 33, 36, 39, 42, 45 and 47 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Extended Static Checking" by Detlefs et al. (hereinafter "Detlefs") in view of U.S. Pat. No. 4,920,538 to Chan et al. (hereinafter "Chan").

With respect to claim 1 (currently amended), Detlefs discloses a method of verifying with static checking whether a specified computer program satisfies a predefined set of conditions (see the title and page 24, Figure 10), comprising:

(a) converting the program into a logical equation representing the predefined set of conditions as applied to instructions and elements of the instructions of the program (see page 23, section 4, paragraph 1, lines 1-4, which shows generating a logical equation based on the conditions annotated in the program).

Although Detlefs discloses that any sub-equation of the logical equation can be given a label (see page 29, section 6, paragraph 2, lines 1-2), Detlefs does not expressly disclose the limitation wherein the converting step includes inserting flow control labels into the sub-

equations of the logical equation, the flow control labels identifying conditional branch points in the specified computer program.

However, Chan discloses inserting control markers or labels into the code to identify conditional branch points (see column 1, lines 38-53), for the purpose of checking execution paths and verifying conditional branches (see column 1, lines 26-35).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the static checking of Detlefs with the labeling feature taught by Chan, for the purpose of verifying conditional branches.

The combination of Detlefs and Chan further discloses:

(b) applying a theorem prover to the logical equation to determine the truth of the logical equation, and when the truth of the logical equation cannot be proved, generating at least one counter-example identifying one of the conditions, one or more variable values inconsistent with the one condition, and any of the flow control labels for conditional branch points of the program associated with the identified variable values (see Detlefs, page 23, section 4, paragraph 1, lines 4-8, which shows using a theorem prover to determine when the truth of the logical equation does not hold, and page 29, section 6, paragraph 2, lines 3-7, which shows generating a counter-example comprising the labels that identify inconsistent conditions in the equation); and

(c) converting the at least one counter-example into an error message that includes a program trace that identifies a path through the computer program when the counter-example identifies one or more of the flow control labels (see Detlefs, page 29, section 6, paragraph 2, lines 3-4 and 7-9, which shows translating a counter-example into an error message that traces

the source of the error using the labels; see also Chan, column 1, lines 26-35, which shows that the labels identify a path through the computer program).

With respect to claim 2 (original), Detlefs further discloses the limitation wherein the converting step additionally comprises a step of converting the program into an intermediate language form prior to creating the logical equation (see page 29, section 6, paragraph 1, lines 7-9, which shows converting the program into Dijkstra's guarded commands, i.e. an intermediate language, before generating the logical equation).

With respect to claim 3 (original), Detlefs further discloses the limitation wherein the flow control labels are inserted before converting the program into the intermediate language form (see page 29, section 6, paragraph 1, lines 7-9, which shows converting an annotated program into an intermediate language; note that annotations are considered labels that are inserted before converting the program).

With respect to claim 4 (original), Detlefs further discloses the limitation wherein the flow control labels are inserted after converting the program into the intermediate language form (see page 29, section 6, paragraph 2, lines 6-7, which shows that the verification condition generator inserts labels into the logical equation; note that because the program is converted into the intermediate language form prior to generating the logical equation, the labels are thereby inserted after converting the program).

With respect to claim 5 (original), Detlefs further discloses the limitation wherein the intermediate language form is Dijkstra's guarded command language (see page 29, section 6, paragraph 1, lines 7-9, which shows using Dijkstra's guarded commands).

With respect to claim 8 (original), the combination of Detlefs and Chan does not expressly disclose the limitation wherein at least one of the flow control labels is in the form L==>P wherein L is a flow control label name and P is a subcomponent of the logical equation.

However, Detlefs discloses using Dijkstra's guarded commands and weakest-precondition equations to generate a verification condition (see page 29, section 6, paragraph 1, lines 7-9). Detlefs further discloses using labels in the verification condition, and using a theorem prover to determine the truth of the logical equation and to find a counter-example (see page 29, section 6, paragraph 2, lines 1-9).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the labels of Detlefs using a mathematical notation such as L==>P. The choice of notation used in the label names does not depart from the teachings of Detlefs, because the functionality remains substantially the same.

With respect to claim 11 (original), the combination of Detlefs and Chan does not expressly disclose the limitation wherein at least one of the flow control labels is in the form L=k==>P wherein L is a flow control label name, k is a constant value and P is a subcomponent of the logical equation.

However, Detlefs discloses using Dijkstra's guarded commands and weakest-precondition equations to generate a verification condition (see page 29, section 6, paragraph 1,

lines 7-9). Detlefs further discloses using labels in the verification condition, and using a theorem prover to determine the truth of the logical equation and to find a counter-example (see page 29, section 6, paragraph 2, lines 1-9).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the labels of Detlefs using a mathematical notation such as L=k==>P. The choice of notation used in the label names does not depart from the teachings of Detlefs, because the functionality remains substantially the same.

With respect to claim 14 (original), the combination of Detlefs and Chan does not expressly disclose the limitation wherein at least one of the flow control labels is in the form {LBLPOS L P} wherein L is a flow control label name and P is a subcomponent of the logical equation.

However, Detlefs discloses using Dijkstra's guarded commands and weakest-precondition equations to generate a verification condition (see page 29, section 6, paragraph 1, lines 7-9). Detlefs further discloses using labels in the verification condition, and using a theorem prover to determine the truth of the logical equation and to find a counter-example (see page 29, section 6, paragraph 2, lines 1-9).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the labels of Detlefs using a mathematical notation such as {LBLPOS L P}. The choice of notation used in the label names does not depart from the teachings of Detlefs, because the functionality remains substantially the same.

With respect to claim 17 (original), the combination of Detlefs and Chan does not expressly disclose the limitation wherein at least one of the flow control labels is in the form ¬{LBLNEG L ¬P} wherein L is a flow control label name and P is a subcomponent of the logical equation.

However, Detlefs discloses using Dijkstra's guarded commands and weakest-precondition equations to generate a verification condition (see page 29, section 6, paragraph 1, lines 7-9). Detlefs further discloses using labels in the verification condition, and using a theorem prover to determine the truth of the logical equation and to find a counter-example (see page 29, section 6, paragraph 2, lines 1-9).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the labels of Detlefs using a mathematical notation such as ¬{LBLNEG L ¬P}. The choice of notation used in the label names does not depart from the teachings of Detlefs, because the functionality remains substantially the same.

With respect to claim 20 (original), the combination of Detlefs and Chan does not expressly disclose the limitation wherein at least one of the flow control labels is in the form {LBLPOS L True}==>P wherein L is a flow control label name and P is a subcomponent of the logical equation.

However, Detlefs discloses using Dijkstra's guarded commands and weakest-precondition equations to generate a verification condition (see page 29, section 6, paragraph 1, lines 7-9). Detlefs further discloses using labels in the verification condition, and using a theorem prover to determine the truth of the logical equation and to find a counter-example (see page 29, section 6, paragraph 2, lines 1-9).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the labels of Detlefs using a mathematical notation such as {LBLPOS L True}==>P. The choice of notation used in the label names does not depart from the teachings of Detlefs, because the functionality remains substantially the same.

With respect to claim 22 (original), Detlefs further discloses the limitation wherein the flow control label name identifies a line number in the specified computer program at which an associated program instruction is located (see page 29, section 6, paragraph 2, lines 7-8, which shows that the name of the label identifies a source position, i.e. a line number in the program).

Detlefs does not expressly disclose the limitation wherein the flow control label includes a sequence number indicating an order of execution of the program instruction at the identified line number relative to other program instructions identified by other flow control labels.

However, Chan further discloses using control markers or labels to identify branches in a program (see column 1, lines 38-53) along with sequence identifiers for each execution path (see column 1, lines 54-57), for the purpose of checking execution paths and verifying conditional branches (see column 1, lines 26-35).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the static checking of Detlefs with the labeling feature taught by Chan, for the purpose of verifying conditional branches.

With respect to claim 23 (currently amended), see the explanation for claim 1 set forth above. Note that Detlefs further discloses that the static checking system is implemented as a computer program product (see page 23, section 4, paragraph 2, lines 1-5). Also note that

Detlefs discloses a verification condition generator, a theorem prover, control labels, and a post processor (see page 24, Figure 10, and page 29, section 6, paragraph 2, lines 1-4).

With respect to claim 24 (original), see the explanation for claim 2 set forth above.

With respect to claim 25 (original), see the explanation for claim 3 set forth above.

With respect to claim 26 (original), see the explanation for claim 4 set forth above.

With respect to claim 27 (original), see the explanation for claim 5 set forth above.

With respect to claim 30 (original), see the explanation for claim 8 set forth above.

With respect to claim 33 (original), see the explanation for claim 11 set forth above.

With respect to claim 36 (original), see the explanation for claim 14 set forth above.

With respect to claim 39 (original), see the explanation for claim 17 set forth above.

With respect to claim 42 (original), see the explanation for claim 20 set forth above.

With respect to claim 45 (original), see the explanation for claim 22 set forth above.

With respect to claim 47 (new), Detlefs discloses a method, comprising:

(a) converting a computer program into a logical equation representing a predefined set of conditions (see page 23, section 4, paragraph 1, lines 1-4, which shows generating a logical equation based on the conditions annotated in the program);

Although Detlefs discloses that any sub-equation of the logical equation can be given a label (see page 29, section 6, paragraph 2, lines 1-2), Detlefs does not expressly disclose:

(b) inserting flow control labels into sub-equations of the logical equation, the flow control labels identifying branch points in the computer program, wherein at least one of the flow control labels is of a form selected from the group consisting of (1) L==>P wherein L is a flow control label name and P is a subcomponent of the logical equation, (2) L=k==22 P wherein L is a flow control label name, k is a constant value and P is a subcomponent of the logical equation, (3) {LBLPOS L P} wherein L is a flow control label name and P is a subcomponent of the logical equation, (4) {LBLNEG L P} wherein L is a flow control label name and P is a subcomponent of the logical equation, and (5) {LBLPOS L True}==>P wherein L is a flow control label name and P is a subcomponent of the logical equation;

However, Chan discloses inserting control markers or labels into the code to identify conditional branch points (see column 1, lines 38-53), for the purpose of checking execution paths and verifying conditional branches (see column 1, lines 26-35).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the static checking of Detlefs with the labeling feature taught by Chan, for the purpose of verifying conditional branches.

Furthermore, Detlefs discloses using Dijkstra's guarded commands and weakest-precondition equations to generate a verification condition (see page 29, section 6, paragraph 1, lines 7-9). Detlefs further discloses using labels in the verification condition, and using a theorem prover to determine the truth of the logical equation and to find a counter-example (see page 29, section 6, paragraph 2, lines 1-9).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the labels of Detlefs using a mathematical notation such as L==>P, L=k==22 P, {LBLPOS L P}, {LBLNEG L P} or {LBLPOS L True}==>P. The choice of notation used in the label names does not depart from the teachings of Detlefs, because the functionality remains substantially the same.

The combination of Detlefs and Chan further discloses:

(c) applying a theorem prover to the logical equation to determine a truth of the logical equation, and when the truth of the logical equation cannot be proved, generating at least one counter-example identifying one of the conditions (see Detlefs, page 23, section 4, paragraph 1, lines 4-8, which shows using a theorem prover to determine when the truth of the logical equation does not hold, and page 29, section 6, paragraph 2, lines 3-7, which shows generating a counter-example comprising the labels that identify inconsistent conditions in the equation); and

(d) converting the at least on counter-example into an error message that comprises a program trace that identifies a path through the computer program when the counter-example identifies one or more of the flow control labels (see Detlefs, page 29, section 6, paragraph 2, lines 3-4 and 7-9, which shows translating a counter-example into an error message that traces the source of the error using the labels; see also Chan, column 1, lines 26-35, which shows that the labels identify a path through the computer program).

9. Claims 6, 7, 9, 10, 12, 13, 15, 16, 18, 19, 21, 28, 29, 31, 32, 34, 35, 37, 38, 40, 41, 43, 44 and 46 are rejected under 35 U.S.C. 103(a) as being unpatentable over Detlefs in view of Chan, as applied to claims 1, 8, 11, 14, 17, 20, 23, 30, 36, 39 and 42 above, respectively, and further in view of U.S. Pat. No. 5,854,924 to Rickel et al. (hereinafter "Rickel").

With respect to claim 6 (original), Detlefs further discloses the limitation wherein at least one of the flow control labels includes a flow control label name that includes a string that identifies a line number in the specified computer program (see page 29, section 6, paragraph 2, lines 7-8, which shows that the name of the label includes a source position, i.e. a line number in the program; note that a label name is inherently a string).

Although Detlefs discloses that the string identifies a type of error (see page 29, section 6, paragraph 2, lines 7-8), Detlefs does not expressly disclose the limitation wherein the string identifies a type of branch in the program.

However, Rickel discloses branch labels in an intermediate language form of a program (see column 8, lines 20-23) used to represent every flow path in the program based on a branch type such as a call or a jump (see column 8, lines 47-53), for the purpose of traversing all the paths in a program with a static debugger and finding any errors (see column 8, lines 61 to column 9, line 3).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the static debugging of Detlefs with the labeling feature taught by Rickel, for the purpose of checking all the paths in a program and finding errors.

With respect to claim 7 (original), although Detlefs discloses a label name that identifies a line number in the computer program (see page 29, section 6, paragraph 2, lines 7-8, which shows that the name of the label identifies a source position, i.e. a line number in the program), Detlefs does not expressly disclose the limitation wherein at least one of the flow control labels

includes a flow control label name that includes a value associated with an entry in a table that identifies a type of branch in the program and a line number in the specified computer program.

However, Rickel further discloses using jump tables and jump targets from the symbol table as branch labels (see FIG. 7), for the purpose of traversing all the paths in a program with a static debugger and finding any errors (see column 8, lines 61 to column 9, line 3).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the static debugging of Detlefs with the labeling feature taught by Rickel, for the purpose of checking all the paths in a program and finding errors.

With respect to claims 9, 12, 15, 18 and 21 (original), see the explanation for claim 6 set forth above.

With respect to claims 10, 13, 16, 19 and 46 (original), see the explanation for claim 7 set forth above.

With respect to claims 28, 31, 37, 40, 43 (original) and 34 (currently amended), see the explanation for claim 6 set forth above.

With respect to claim 29, 32, 38, 41, 44 (original) and 35 (currently amended), see the explanation for claim 7 set forth above.

Conclusion

10. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

11. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (703) 305-0352. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (703) 305-4552. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

MY

Michael J. Yigdall
Examiner
Art Unit 2122

mjy
May 11, 2004

TUAN DAM
SUPERVISORY PATENT EXAMINER